Algorithm executed by a Master-designate

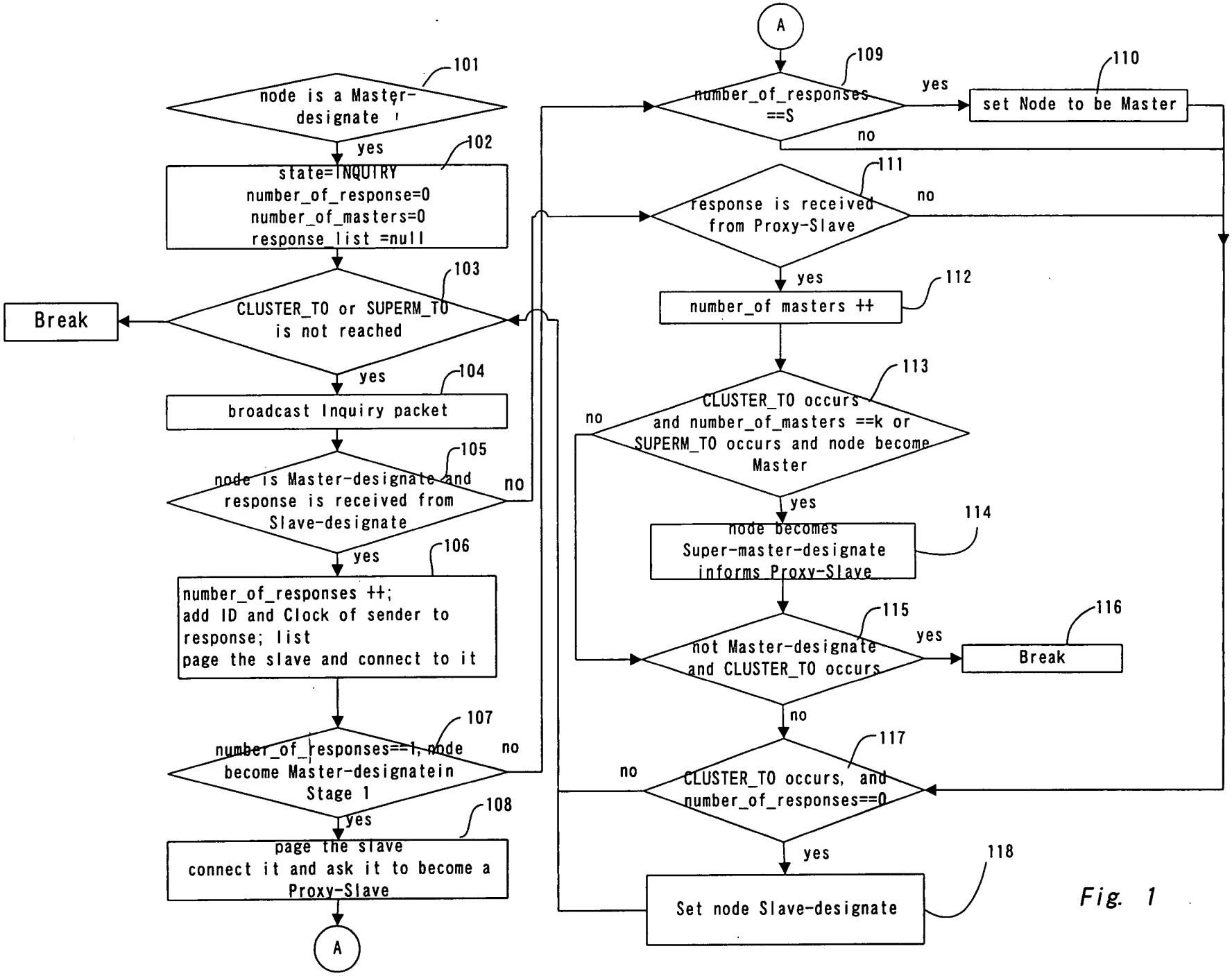


Fig. 1

Algorithm executed by a Master-designate:

```
if ( node is a Master-designate or Master ) then
        state = INQUIRY;
        number_of_responses =0;
        number_of_masters=0;
        response_list = nil;
        while ( CLUSTER_TO is not reached or SUPERM_TO is not reached )
            broadcast Inquiry packet;
            if ( node is Master-designate and response is received from Slave_designate)
then
                number_of_responses++;
                add Id and Clock of sender to response_list;
page the slave and connect to it;
                if ( number_of_responses == 1 and node has become Master-designate by
stage-1) then
                    page the slave, connect to it and ask it to become a Proxy-slave;
                if ( number_of_responses == S ) then Node becomes Master;
            else if ( response is received from Proxy-slave ) then
                number_of_masters++;
                if ( CLUSTER_TO occurs and (number_of_masters == k or
                    SUPERM_TO occurs) and node has become Master-designate by stage-1)
then
                        node becomes Super-master-designate;
                        informs Proxy-slave;
                if ( node has not become Master-designate by stage-1 and CLUSTER_TO
occurs ) then
                        Break;
            if ( CLUSTER_TO occurs and number_of_responses == 0 ) then
                // node becomes Slave-designate
        endwhile
endif
```

Fig. 2

Algorithm executed by a Slave-designate

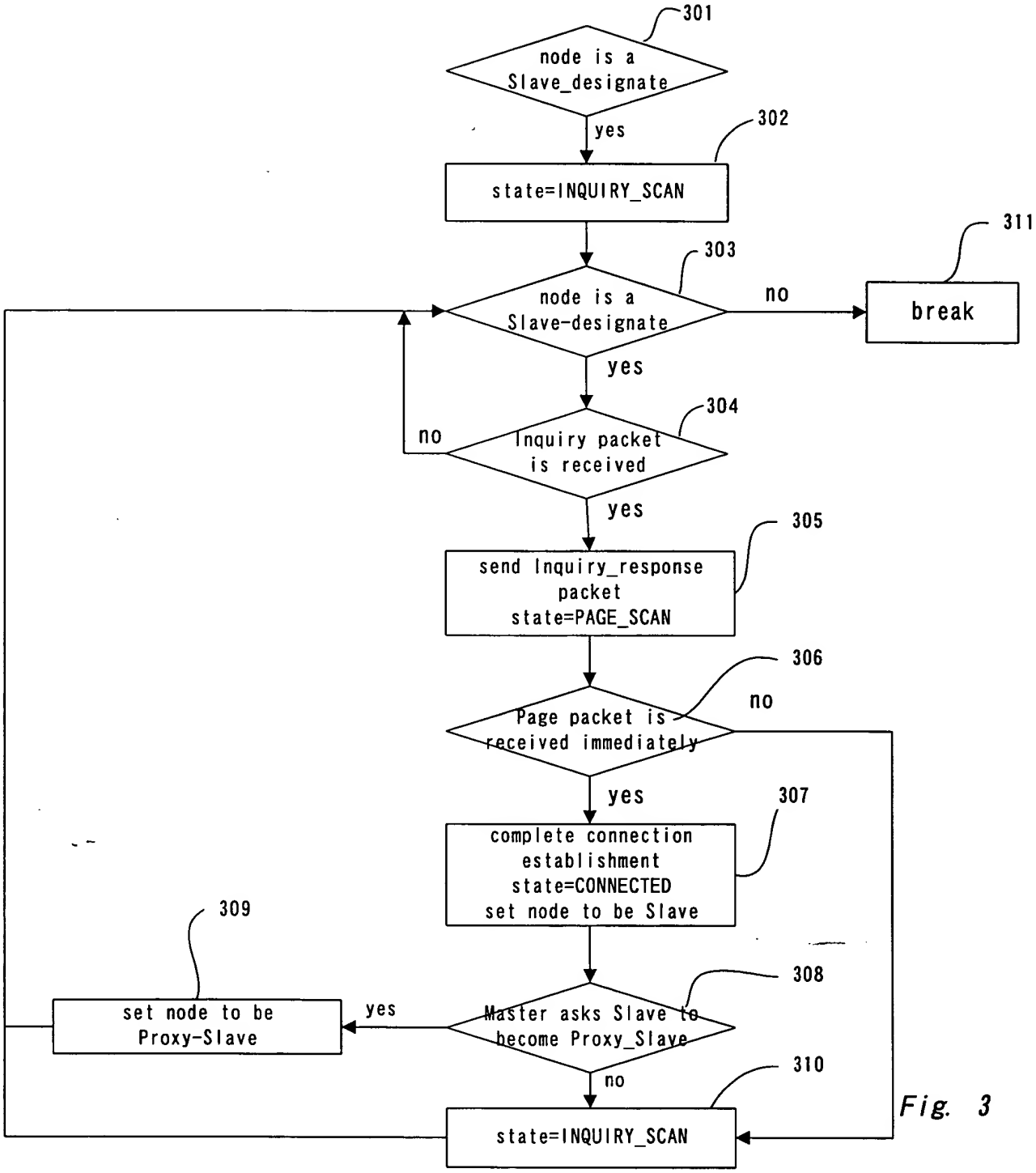


Fig. 3

Algorithm executed by a Slave-designate:

```
if ( node is a Slave_designate ) then
        state = INQUIRY_SCAN;
        while ( node is a Slave-designate )
            if ( Inquiry packet is received ) then
                send Inquiry_response packet ;
                  state = PAGE_SCAN;
                  if ( Page packet is received immediately) then
                      complete connection establishment;
                      state = CONNECTED;
                      Node becomes Slave;
                       if ( Master asks it to become Proxy_slave ) then
                              node becomes Proxy-slave;
                  else // it is not paged
                      state = INQUIRY_SCAN;
        endwhile
endif
```

Fig. 4

Algorithm executed by a Proxy-Sl

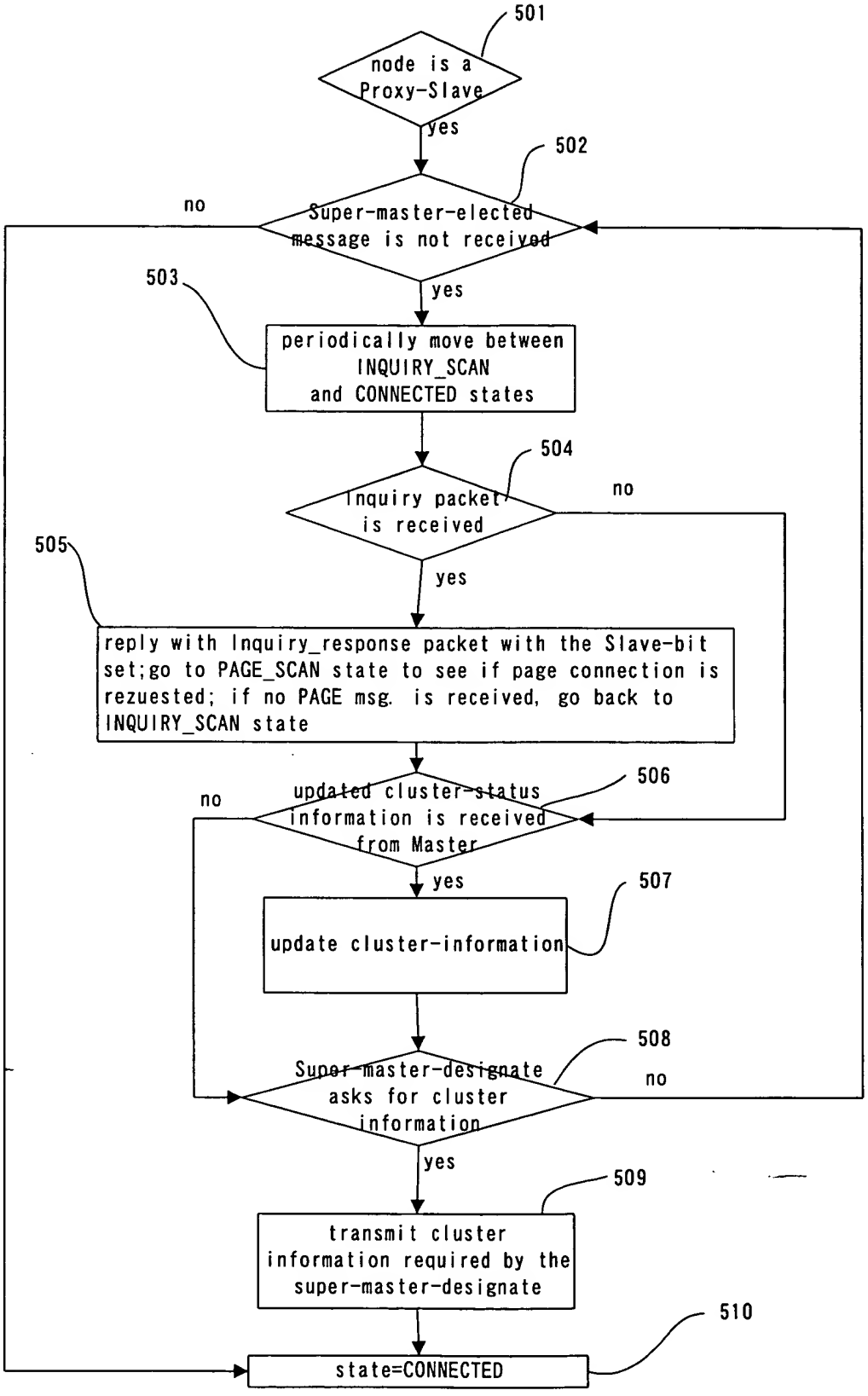


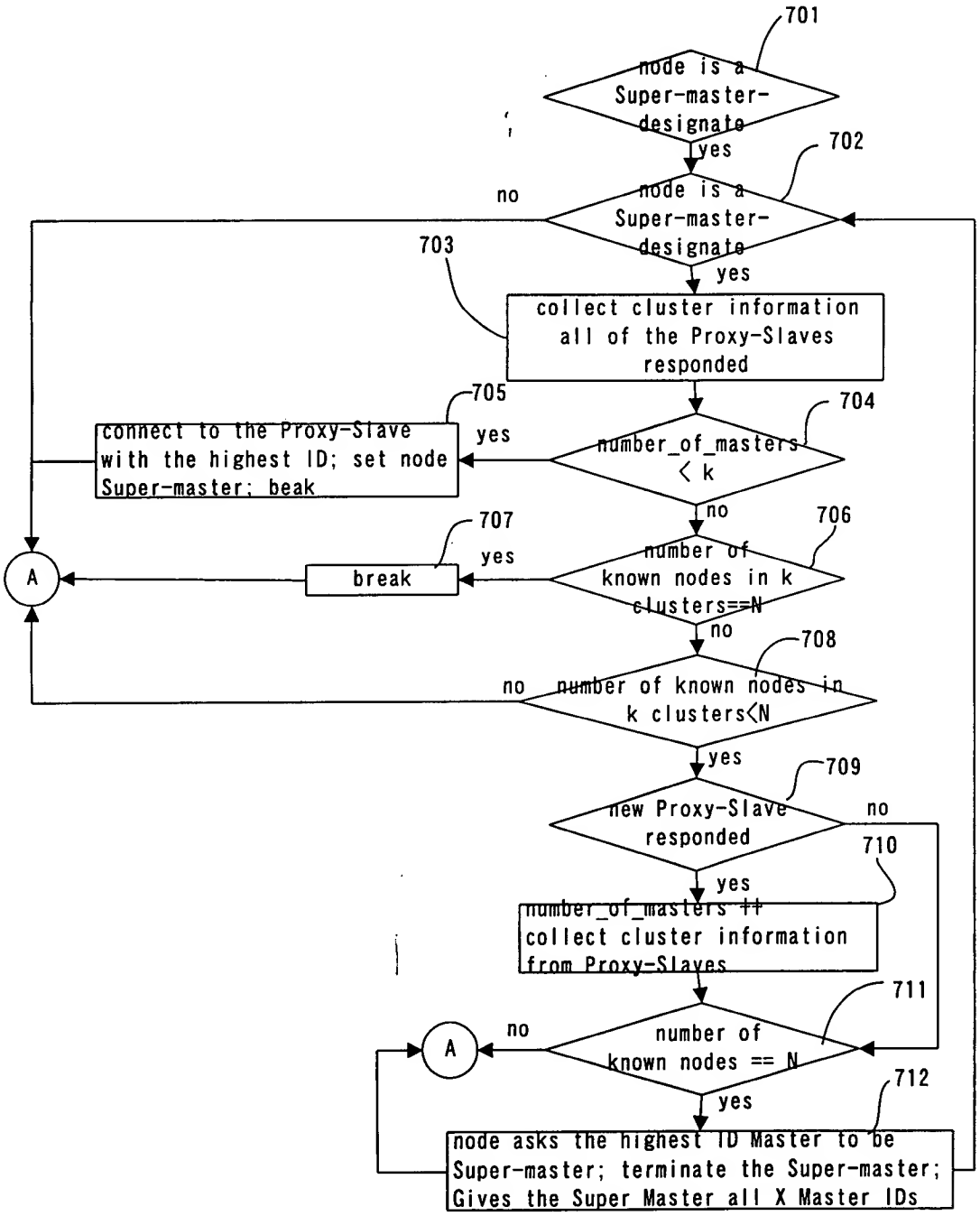
Fig. 5

Algorithm executed by a Proxy-slave:

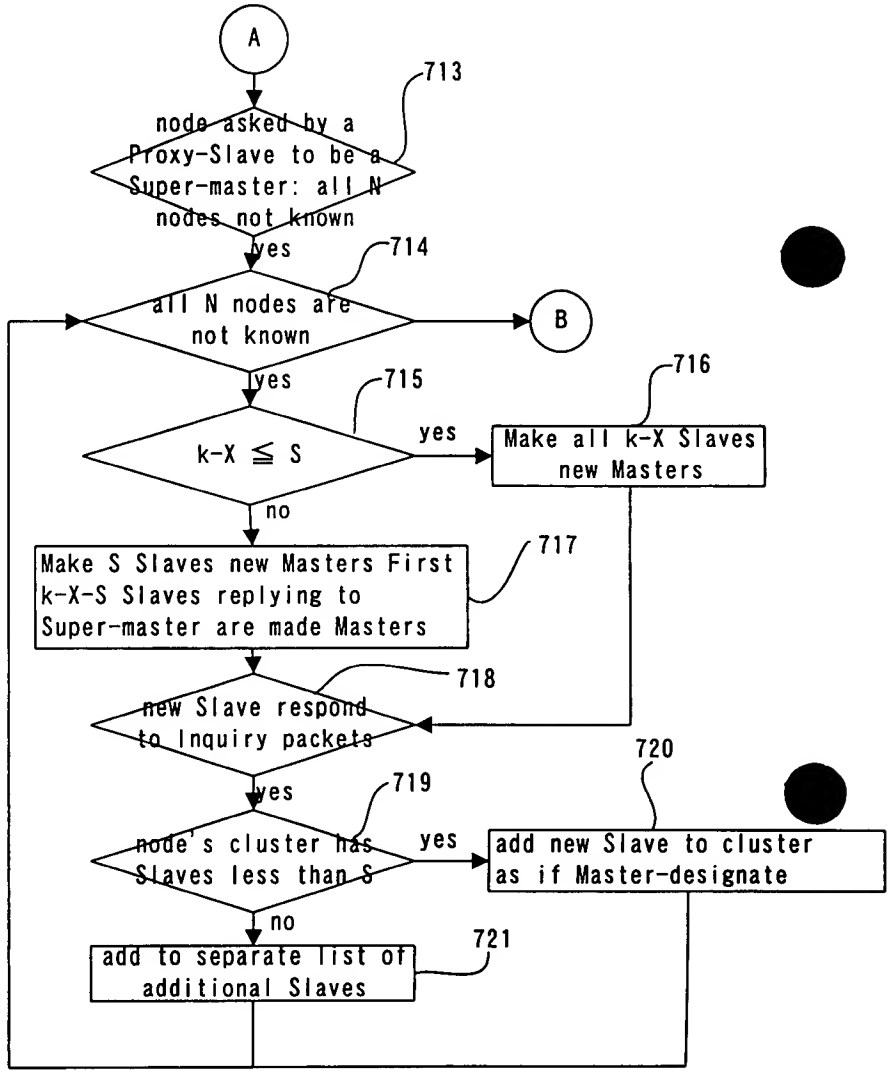
```
if ( node is a Proxy-slave ) then
        // node knows Master ID
        // this node is in CONNECTED state
        while ( Super-master-elected message is not received )
             periodically move between INQUIRY_SCAN
             and  CONNECTED states;
             // the amount of time in inq-scan being much more than that in connected
             if ( Inquiry packet is received ) then
                // state is in INQUIRY_SCAN
                reply with Inquiry_response packet with the Slave-bit set;
                go to PAGE_SCAN state to see if page connection is requested;
                if no Page msg is received, go back to INQUIRY_SCAN state;
            if ( updated cluster-status information is received from Master ) then
                update cluster-information;
            if ( super-master-designate asks for cluster information ) then
                // temporarily becomes a bridge between its own cluster
                // and that between itself and the Super-master-designate
                send cluster information required by the Super-master-designate;
        endwhile
        // super-master election is complete
        state = CONNECTED;
endif
```

Fig. 6

Algorithm executed for Super-master-election



(a)



(b)

Fig. 7

Algorithm executed for Super-master-election (2)

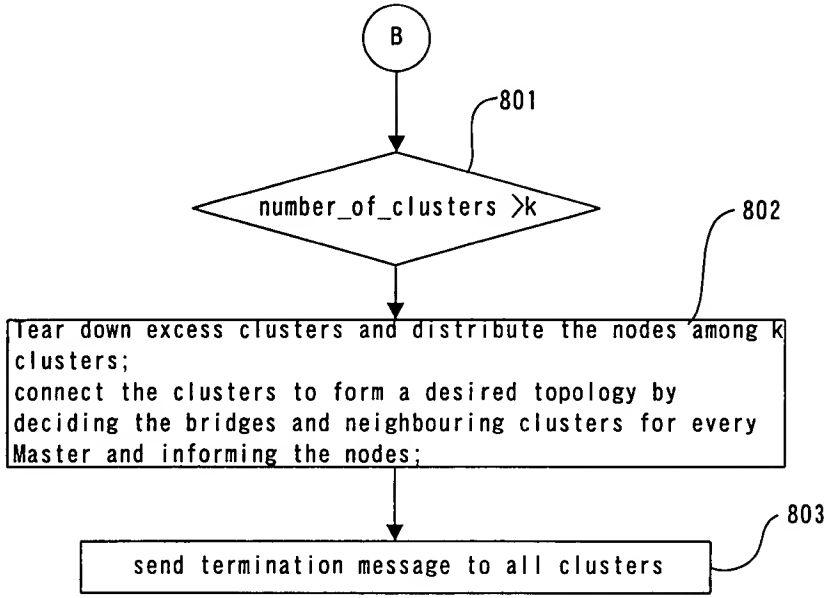
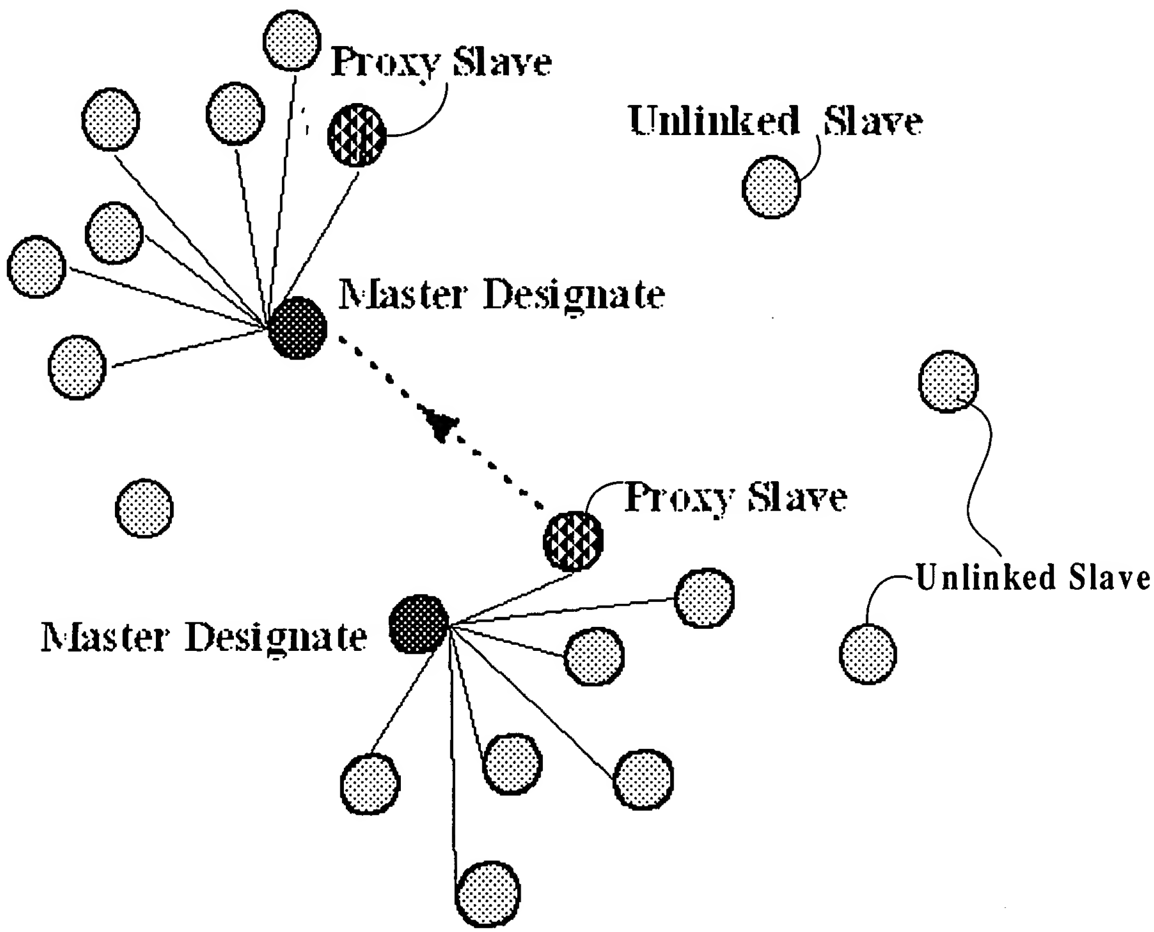


Fig. 8

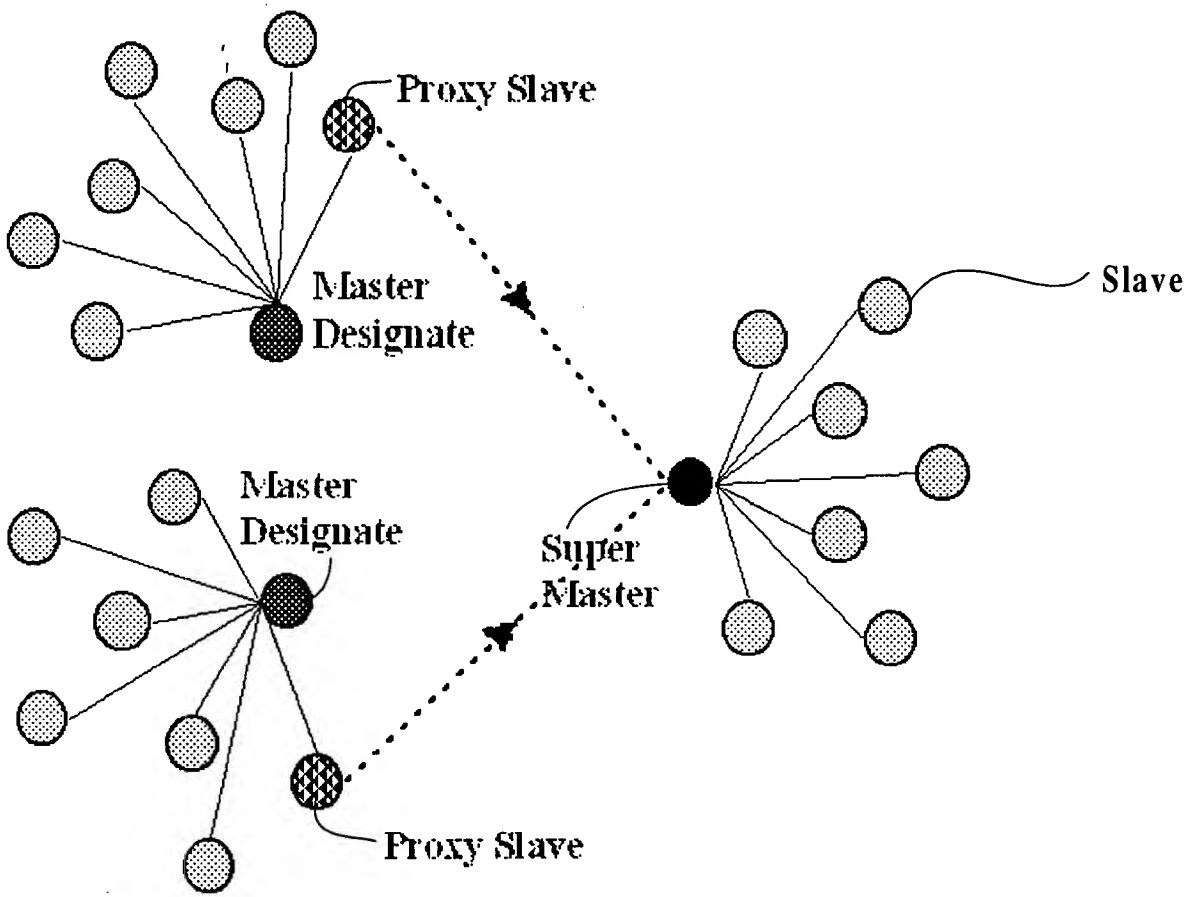
```
Algorithm executed for Super-master-election:
if  ( node is a Super-master-designate ) then
    while ( node is Super-master-designate )
       collect cluster information from all the Proxy-slaves which responded;
       // in order to make a connection to a Proxy-slave, send Inquiry pkt, so
that it goes to
       // page-scan soon after responding and connection can be established.
       if ( number_of_masters < k ) then
              // SUPERM_TO has occurred
              make a connection to the Proxy-slave with the highest Id
              node becomes Super-master;
              break;
         // number of masters = k ( actual number-of-masters > k )
          if ( total number of known nodes in k clusters == N ) then // X = k
              break;
               if ( total number of known nodes in k clusters < N ) then // X > k
             if ( new Proxy-slaves respond to Inquiry packets ) then
              number_of_masters++ ;
              collect cluster info. from them;
             if ( total number of known nodes == N ) then
              node asks the highest Id Master to become Super-master;
              also tells the Super-master to terminate;
              Gives the Super-master all X Master Ids;
  endwhile
   if ( node has been asked by a Proxy-slave to become Super-master and
        all N nodes not known )  then // X < k
     while( all N nodes are not known )
       if ( k-X <= S ) then
           Make all k-X slaves new Masters, which then collect the remaining
nodes;
       else
           Make S slaves new Masters, which collect new Slave-designates
           The first k-X-S Slave-designates that reply to it are made Masters;
            if ( new Slave-designates respond to Inquiry packets ) then
              // these are orphan slave-designates which are not part of any
cluster
              if ( the node*s cluster has < S slaves ) then
                  add new slave to cluster as if Master-designate ;
              else
                  add to separate list of additional slaves ;
    endwhile
    if ( number_of_clusters > k ) then
       Tear down excess clusters and distribute the nodes among k clusters;
       connect the clusters to form a desired topology by deciding the
       bridges and neighbouring clusters for every Master, and informing the
       nodes;
   send termination message to all clusters ( and hence all nodes );
endif
```

Fig. 9



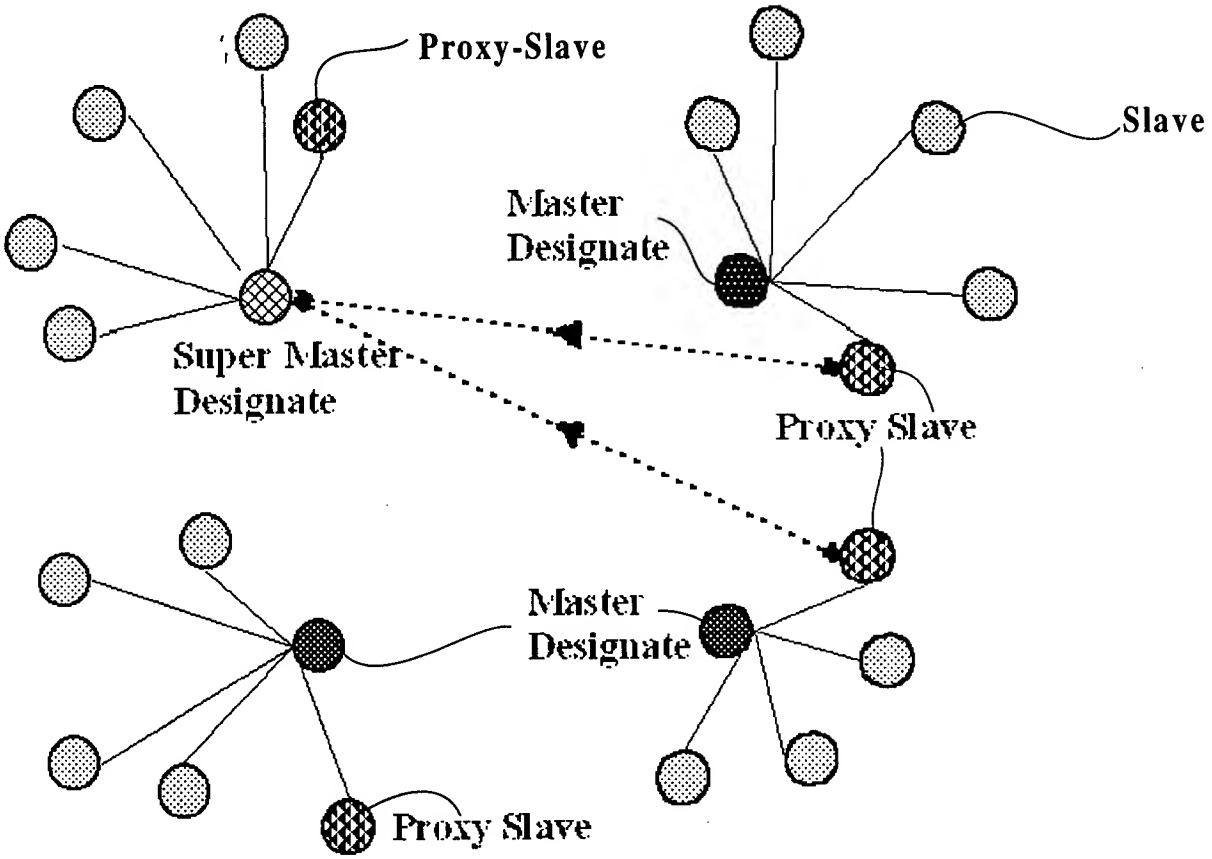
Randomized cluster formation algorithm for X<k

Fig. 10



Randomized cluster formation algorithm for X=k

Fig. 11



Randomized cluster formation algorithm for X>k

Fig. 12